Return to article page

PR Newswire
April 24, 2000

Financial Fusion, Inc. Collaborates with IBM on 'Project Indigo'.

Strategic Initiative to Map Financial Fusion's Web, Wireless and Server

Solutions to IBM's Application Framework for E-Business

WESTPORT, Conn., April 24 /PRNewswire/ --

Financial Fusion, Inc., announced today "Project Indigo," a joint initiative to build seamless connectivity between Financial Fusion's Web, wireless and server applications and IBM's Application Framework for e-Business. This extends the existing relationship between the two companies and further assists financial services companies as they accelerate their e-business initiatives and move to offer a greater range of products and services to consumers via the Web.

The Project Indigo solution suite will integrate Financial Fusion's Web, wireless and server products with offerings within IBM's Application Framework for e-Business, including MQSeries, AIX, DB2 Universal Database and WebSphere application server family, as well as S/390 enterprise server platform. As a result, the companies will be able to offer Global 1000 financial institutions a broad suite of e-finance solutions all on a single, integrated platform that are flexible, scalable, quick to deploy and highly secure. The Project Indigo solution suite will be promoted and marketed globally by IBM and Financial Fusion with integration services and support to be provided primarily by IBM Global Services, as well as other leading global system integration firms.

In alignment with Financial Fusion's and IBM's core business strategies, the Project Indigo solution suite will provide e-finance solutions based on existing industry standards and built on open platforms to help developers integrate Internet technologies with traditional information technology. Financial Fusion's Web and wireless applications, which include full transaction functionality for e-banking, bill payment and bill presentment, as well as content management services and relationship marketing tools, are all developed using Java, XML, Corba and other standard, cross-

platform technologies. Financial Fusion Server, also developed in Java and XML, is rapidly becoming the de facto standard for financial middleware and offers seamless connectivity for the retail delivery and capital markets sectors via industry standard protocols including OFX, FIX and S.W.I.F.T. Project Indigo solutions will also be truly global, offering double-byte, multi-currency and National Language support.

"Project Indigo is a 'pure blue' initiative designed to enhance our product offerings with a series of powerful, versatile e-finance solutions that leverage IBM's global relationships with the world's leading financial institutions," said Daniel M. Schley, CEO of Financial Fusion. "We have long recognized that to achieve this global mission we would need to develop a technology roadmap resulting in the seamless integration of our front-end applications and middleware technology with IBM's hardware/software offerings. Today, we are well down the path to completing our integration efforts and have already deployed solutions at an international bank which supports MQSeries and in a technology licensing agreement which is based on AIX."

"Banks and other financial services institutions have to transform themselves into global e-businesses, if they are to have a significant share of their customers' total financial transactions," said Michael Cannon-Brookes, general manager, IBM Global Banking Industry. "Our e-business expertise and global presence, combined with Financial Fusion's mastery in building consumer-friendly online delivery channels for financial services are a winning combination for our joint customers."

IBM's Application Framework for e-Business is a technological roadmap, based on industry standards, that helps developers integrate Internet technology with traditional information technology. IBM WebSphere application server family, MQ Series messaging middleware and DB2 Universal Database servers are among the strategic products within its Application Framework for e-Business.

About Financial Fusion, Inc. (http://www.financialfusion.com)

Financial Fusion, Inc. is a single source, global provider of e-finance web, wireless and server solutions targeting Global 1000 financial institutions. Through the company's Financial Fusion System and Java-based Stage III Architecture, Financial Fusion provides the enabling technology to build fully functional "financial communities" on the web, provide seamless connectivity on any internet-enabled device, and serve both the retail and capital markets sectors -- all on a single, integrated platform. Financial Fusion's global alliance partners include IBM, Sun Microsystems, Sanchez, Intuit and CheckFree. Financial Fusion, Inc. is a wholly owned subsidiary of Sybase, Inc. (Nasdaq: SYBS).

L4: Entry 2 of 3 File: USPT Jul 8, 2003

US-PAT-NO: 6591272
DOCUMENT-IDENTIFIER: US 6591272 B1

TITLE: Method and apparatus to make and transmit objects from a database on a server computer to a client computer

DATE-ISSUED: July 8, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Williams; Mark	Capitola	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Tricoron Networks, Inc.	Capitola	CA			02

APPL-NO: 09/ 510750 [PALM]
DATE FILED: February 22, 2000

PARENT-CASE:
This application claims the benefit of U.S. Provisional Application No. 60/121,527, filed on Feb. 25, 1999, entitled "Expert System to Build and Deploy Coherent Networked Distributed Objected Applications".

INT-CL: [07] G06 F 7/00, G06 F 17/00

US-CL-ISSUED: 707/102; 707/100, 707/101
US-CL-CURRENT: 707/102; 707/100, 707/101

FIELD-OF-SEARCH: 707/1-10, 707/100-104.1, 706/50, 345/763

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4930071	May 1990	Tou et al.	706/50
☐	5295256	March 1994	Bapat	717/137
☐	5418717	May 1995	Su et al.	704/9
☐	5542078	July 1996	Martel et al.	707/101
☐	5627979	May 1997	Chang et al.	345/763

h e b b g e e e f c e b e ge

☐	5630131	May 1997	Palevich et al.	717/108
☐	5652884	July 1997	Palevich	713/1
☐	5809505	September 1998	Lo et al.	707/100
☐	5832498	November 1998	Exertier	707/1
☐	5857191	January 1999	Blackwell et al.	707/10
☐	5857197	January 1999	Mullins	707/103
☐	5893108	April 1999	Srinivasan et al.	707/103R
☐	5956725	September 1999	Burroughs et al.	707/100
☐	5970490	October 1999	Morgenstern	707/10
☐	5999734	December 1999	Willis et al.	717/149
☐	6038565	March 2000	Nock	707/101
☐	6061515	May 2000	Chang et al.	707/100
☐	6175837	January 2001	Sharma et al.	707/102
☐	6374256	April 2002	Ng et al.	707/100
☐	6453312	September 2002	Goiffon et al.	707/3

OTHER PUBLICATIONS

"Relational Data Hits the Highway", Julia K. Miller and Thomas Kern, Distributed Computing, Jul. 1998, pp. 38-42.

ART-UNIT: 2177

PRIMARY-EXAMINER: Robinson; Greta

ASSISTANT-EXAMINER: Black; Linh

ATTY-AGENT-FIRM: Gray Cary Ware & Freidenrich LLP

ABSTRACT:

Contents of databases are translated into objects by reading the database schema metadata to determine data interrelationships and create objects with nominal human to computer interaction. Metadata for any number of databases is normalized in a standardized view. Skeleton code templates representative of final classes to be produced are accessed and merged with the standardized view. Source code for the class of the objects is then generated. At runtime, data objects are then produced by encapsulating the metadata and data values. Communication between database instances and a client computer consists of metadata and database row values., Rows from database tables and the corresponding metadata are transmitted from the server to the client computer in one logical network operation. The final distributed objects are then assembled into the optimal format required by the client computer. To update, delete or create new persistent objects, the reverse process occurs.

8 Claims, 22 Drawing figures

Previous Doc Next Doc Go to Doc#

h e b b g e e e f c e b e ge

L4: Entry 2 of 3 File: USPT Jul 8, 2003

US-PAT-NO: 6591272
DOCUMENT-IDENTIFIER: US 6591272 B1

TITLE: Method and apparatus to make and transmit objects from a database on a server computer to a client computer

DRAWING DESCRIPTION:

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a schematic hardware view of the relationship of a client computer and a server computer in the environment in which the methods of the present invention are utilized.

FIG. 2 is a schematic software view of an embodiment of one method of the present invention with regard to the translation of elements of a relational database table into classes desired.

FIG. 3 is a schematic block level diagram of a software view of another method of the present invention in communicating elements of a database tables from a one or more database server computers and/or middle-tier computers to a client computer.

FIG. 4 is a chart showing Normalization Object Topology.

FIG. 5 is a chart showing OSF Object Hierarchy.

FIG. 6 is a model of Basic OSF Persistence.

FIG. 7 is a screen shot of an example of a Generated HTML Page.

FIG. 8 is a screen shot of an example of a Non-Frames Generated HTML Page.

FIG. 9 is a screen shot of a Database Connect Panel--DB Login.

FIG. 10 is a screen shot of a Database Connect Panel--Advanced Connect.

FIG. 11 is a screen shot of a Database Connect Panel--DB Driver and URL.

FIG. 12 is a screen shot of a Database and Table Select--No Selections Made.

FIG. 13 is a screen shot of a Database and Table Select--CUSTOMER Table Selected.

FIG. 14 is a screen shot of an Object and Attribute Naming.

FIG. 15 is a screen shot of an Object and Attribute Naming--Attribute Name Correction.

FIG. 16 is a screen shot of a Database and Table Select--All Tables Selected.

FIG. 17 is a screen shot of a Generation Options--Primary Options.

FIG. 18 is a screen shot of a Generation Options--Enterprise Architecture.

FIG. 19 is a screen shot of a Generation Options--Secondary Options.

FIG. 20 is a graph of an OSF High Level Input Output.

FIG. 21 is a screen shot of an IDE Design Mode HTML. Template.

FIG. 22 is a screen shot of a Generated HTML Subframe.

Previous Doc Next Doc Go to Doc#

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 3 of 3 returned.

1. Document ID: US 6640145 B2

L1: Entry 1 of 3 File: USPT Oct 28, 2003

US-PAT-NO: 6640145
DOCUMENT-IDENTIFIER: US 6640145 B2

TITLE: Media recording device with packet data interface

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw De

2. Document ID: US 6591272 B1

L1: Entry 2 of 3 File: USPT Jul 8, 2003

US-PAT-NO: 6591272
DOCUMENT-IDENTIFIER: US 6591272 B1

TITLE: Method and apparatus to make and transmit objects from a database on a server computer to a client computer

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw De

3. Document ID: US 5970490 A

L1: Entry 3 of 3 File: USPT Oct 19, 1999

US-PAT-NO: 5970490
DOCUMENT-IDENTIFIER: US 5970490 A

TITLE: Integration platform for heterogeneous databases

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw De

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Terms	Documents
6640145.pn. or 6591272.pn. or 5970490.pn.	3

h e b b g e e e f e b ef b e

Display Format: TI Change Format

Previous Page Next Page Go to Doc#

h e b b g e e e f e b ef b e

L1: Entry 2 of 3 File: USPT Jul 8, 2003

US-PAT-NO: 6591272
DOCUMENT-IDENTIFIER: US 6591272 B1

TITLE: Method and apparatus to make and transmit objects from a database on a server computer to a client computer

DATE-ISSUED: July 8, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Williams; Mark	Capitola	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Tricoron Networks, Inc.	Capitola	CA			02

APPL-NO: 09/ 510750 [PALM]
DATE FILED: February 22, 2000

PARENT-CASE:
This application claims the benefit of U.S. Provisional Application No. 60/121,527, filed on Feb. 25, 1999, entitled "Expert System to Build and Deploy Coherent Networked Distributed Objected Applications".

INT-CL: [07] G06 F 7/00, G06 F 17/00

US-CL-ISSUED: 707/102; 707/100, 707/101
US-CL-CURRENT: 707/102; 707/100, 707/101

FIELD-OF-SEARCH: 707/1-10, 707/100-104.1, 706/50, 345/763

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4930071	May 1990	Tou et al.	706/50
☐	5295256	March 1994	Bapat	717/137
☐	5418717	May 1995	Su et al.	704/9
☐	5542078	July 1996	Martel et al.	707/101
☐	5627979	May 1997	Chang et al.	345/763

h e b b g e e e f c e b e ge

☐	5630131	May 1997	Palevich et al.	717/108
☐	5652884	July 1997	Palevich	713/1
☐	5809505	September 1998	Lo et al.	707/100
☐	5832498	November 1998	Exertier	707/1
☐	5857191	January 1999	Blackwell et al.	707/10
☐	5857197	January 1999	Mullins	707/103
☐	5893108	April 1999	Srinivasan et al.	707/103R
☐	5956725	September 1999	Burroughs et al.	707/100
☐	5970490	October 1999	Morgenstern	707/10
☐	5999734	December 1999	Willis et al.	717/149
☐	6038565	March 2000	Nock	707/101
☐	6061515	May 2000	Chang et al.	707/100
☐	6175837	January 2001	Sharma et al.	707/102
☐	6374256	April 2002	Ng et al.	707/100
☐	6453312	September 2002	Goiffon et al.	707/3

OTHER PUBLICATIONS

"Relational Data Hits the Highway", Julia K. Miller and Thomas Kern, Distributed Computing, Jul. 1998, pp. 38-42.

ART-UNIT: 2177

PRIMARY-EXAMINER: Robinson; Greta

ASSISTANT-EXAMINER: Black; Linh

ATTY-AGENT-FIRM: Gray Cary Ware & Freidenrich LLP

ABSTRACT:

Contents of databases are translated into objects by reading the database schema metadata to determine data interrelationships and create objects with nominal human to computer interaction. Metadata for any number of databases is normalized in a standardized view. Skeleton code templates representative of final classes to be produced are accessed and merged with the standardized view. Source code for the class of the objects is then generated. At runtime, data objects are then produced by encapsulating the metadata and data values. Communication between database instances and a client computer consists of metadata and database row values., Rows from database tables and the corresponding metadata are transmitted from the server to the client computer in one logical network operation. The final distributed objects are then assembled into the optimal format required by the client computer. To update, delete or create new persistent objects, the reverse process occurs.

8 Claims, 22 Drawing figures

Previous Doc Next Doc Go to Doc#

h e b b g e e e f c e b e ge

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#
End of Result Set

Generate Collection Print

L10: Entry 2 of 2 File: USPT Jul 8, 2003

DOCUMENT-IDENTIFIER: US 6591272 B1
TITLE: Method and apparatus to make and transmit objects from a database on a server computer to a client computer

Detailed Description Text (25):
OSF Support Classes are then discussed. Examples of these support classes include pick list generation, distributed edit/business rules, and real-time performance measurement and analysis. The Registry class is central to runtime system configuration and it is described in this section.

Detailed Description Text (255):
(23) Ultrathin Client Architecture Rule #1 in the distributed component business is to "Never let the users fall asleep in front of their workstations".

Previous Doc Next Doc Go to Doc#

h e b b g e e e f c e b e ge

L7: Entry 2 of 3 File: USPT Jul 8, 2003

DOCUMENT-IDENTIFIER: US 6591272 B1
TITLE: Method and apparatus to make and transmit objects from a database on a server computer to a client computer

Drawing Description Text (10):
FIG. 9 is a screen shot of a Database Connect Panel--DB Login.

Detailed Description Text (94):
A comprehensive exception handling scheme handles all server-side exceptions, standardizes and normalizes them then transmits the exceptions via CORBA. When received at the client end or requesting server-side middleware, PersistentobjectEvent.COMPONENTEXCEPTION events are fired to all registered listeners in the PRO-OBJECT with all indicative data about the exception in a format presentable to the end-user.

Detailed Description Text (118):
Shared Activation Mode: Shared activation mode can be used when registering a CORBA object server implementation since shared activation mode saves memory and nominalizes ORB overhead. Since all object servers, regardless of architecture, start a thread each time a database has to be accessed, one user will not affect another in the server in shared activation mode. Per-client activation mode can also be used if lots of server resources are available and the absolute best performance is desired for the client workstations/end users. We recommend this option and it is the default used in the script that registers CORBA object servers with the ORB.

Detailed Description Text (135):
Then only the attributes that are to be changed in the persistent relational object are added to the OSFORBStream. In addition to the attribute ID and the new attribute value, the old attribute value is added to the OSFORBStream as well. Given that PRO-OBJECTS can take the form of JavaBean components, it makes sense to handle the persistent relational update in the same manner as the update of a JavaBean bound property (in fact, that'is precisely what occurs: the attribute property is changed and then the remote RDB is synchronized, with the old, previous value of the attribute being sent to the server in the OSFORBStream). The OSFORBStream is then transmitted to the server implementation. A remote server exception will restore any changes made to bound properties and fire a PersistentObjectEvent.COMPONENTEXCEPTION to all registered event listeners.

Detailed Description Text (138):
These steps are taken if the attribute value as believed current by the client is not matched to the column value in the database: A rollback( ) is issued against the current Connection object in the server implementation to roll out any partially completed updates and to free all locks an OSFDBValueUpdateCompareException is thrown over CORBA to the client PRO-OBJECT a COMPONENTEXCEPTION PersistentObjectEvent is thrown in the PRO-OBJECT to all interested and registered event listeners the end user notified that he or she was dealing with stale data

Detailed Description Text (234):

h e b b g e e e f c e b e ge

The solution is simple. Create another WWW server, install the servlet .class files, register the servlets and configure the servlet.properties and other properties needed by the web server and test.

Detailed Description Text (253):
Many applications built today require a user to terminate and restart the application when a network, hardware or software failure occurs. Also users may have to logoff and login/reauthenticate when a network, hardware or software problem occurs. We consider both of these methods of human, end-user recovery to be not at all acceptable.

Detailed Description Text (254):
Each ObjectServerFactory architecture solution offers transparent recovery in the event of network, hardware or server software component failure. In addition, server load can be easily balanced between servers within a given login session. How this capability is enabled through solid design is and intelligent design patterns are outlined in the following sections.

Detailed Description Text (306):
In addition to client-end and server-side persistent, relational object classes, OSF generates: OMG Interface Definition Language which exposes remote server methods to PRO-OBJECT based clients Build scripts for all generated code, including invocation of the IDL compiler and compiling IDL output A server registration script to register the CORBA server implementations with the Object Request Broker Master sedlanguage translation scripts to propagate translations to the various java.util.ListResourceBundle-derived objects HTML template files for data entry, inquiry and tabular display A Registry.java file containing all runtime parameters for a given installation, along with accessor classes and the object map Test programs for standalone testing of PRO-OBJECT component Other assorted utility and convenience scripts including a buildall script which builds everything in the proper sequence, interleaving builds into separate processes when possible

Detailed Description Text (402):
The Database Connect window contains three property pages used to enter the parameters needed to connect to the various relational databases: DB Login, Advanced Connect and Drivers/URL.

Detailed Description Text (403):
(39)DB Login Panel

Detailed Description Text (705):
This section is reserved for important classes not built by OSF but used to support the various runtime environments. Examples of these classes are: The Registry class contains all of the parameters which are unique to a given customer application. Database connect parameters and driver information, default database server IP addresses, initial object-> base table and column mapping parameters and basic rules edit parameters are contained in the Registry class. Also, a few parameters that were initially manifest constants were moved out of the code into the Registry so the values could be changed without recompiling the application modules. The OSFControlServlet class is the servlet that invokes the OSFSecurity object to validate logins, perform runtime authorization and to switch the browser context from servlet to servlet. OSFPickListBuildThread is the class that scans each database table to construct default edit rules and to build lists of possible pick list candidates.

Previous Doc Next Doc Go to Doc#

L5: Entry 1 of 2 File: USPT Jul 8, 2003

DOCUMENT-IDENTIFIER: US 6591272 B1
TITLE: Method and apparatus to make and transmit objects from a database on a server computer to a client computer

Detailed Description Paragraph Table (80):
public void appenAttribute (String_attribute) { // if a delimiter is encountered, add an ajacent delimiter which // will be removed by the ORB Stream parser on the receiving end // null checks first if (_attribute == null) { append (NULLATTRIBUTE) ; return; } if (_attribute.length ( ) == 0) { append (NULLATTRIBUTE) ; return; } // scan for delims, if found replace delimiter _attribute.replace (STREAMDELIMITER, DELIMITERINSTREAM) ; append (_attribute) ; }

Detailed Description Paragraph Table (107):
public string getParameter(String _searchargument) throws IllegalArgumentException { String parameter = (string) table.get(_searchargument); if (parameter == null) { displayErrorMessage (_searchargument); throw new IllegalArgumentException("Not found in registry: " + _searchargument); } return parameter; } public String getParameterAsString (String _searchargument) throws IllegalArgumentException { return getParameter(_searchargument); } public int getParameterAsInt (String _searchargument) throws IllegalArgumentException { String parameter = (String) table.get(_searchargument); if (parameter == null) { displayErrorMessage (_searchargument) throw new IllegalArgumentException("Not found in registry: " + _searchargument); } int parsedparameter = 0; try { parsedparameter = Integer.parseInt(parameter); } catch (NumberFormatException nfe) { throw new IllegalArgumentException("Registry arg format error, not an int-" + parameter + ", s-" + _searchargument); } return parsedparameter; } public double getParameterAsDouble (String _searchargument) throws IllegalArgumentException { String parameter = (String) table.get(_searchargument); if (parameter == null) { displayErrorMessage (_searchargument); throw new IllegalArgumentException("Not found in registry: " + _searchargument); } double parsedparameter = 0.0; try { parsedparameter = Double.valueOf(parameter).doubleValue( ); } catch (NumberFormatException nfe) { String message = "Registry arg format error, not a double-" + parameter + ", s-" + searchargument; throw new IllegalArgumentException (message); } return parsedparameter; } public boolean getParameterAsBoolean(String _searchargument) throws IllegalArgumentException { String parameter = (string) table.get(_searchargument); if (parameter == null) { displayErrorMessage (_searchargument); throw new IllegalArgumentException("Not found in registry: " + _searchargument); } if (parameter.compareTo(TRUE) == 0) { return true; } return false; } private void displayErrorMessage (String _searchargument) { // build message String message = this.getClass( ).getName( ) + "-E-NotInRegistry, paramater with internal representation-" + _searchargument + "not found in Registry"; // log message using sysman ref if available if (sysman_ != null) { sysman_.logMessage (message); } else { System.out.println(message); } } }

Detailed Description Paragraph Table (111):
Target in Skeleton Function and Operation by Template file OSFGenerate ##Package## -> package target (0) ##TableObjectName## -> normalised table name (1) ##TABLENAME## -> insert table name in UPPER CASE (2) ##COLUMNNAMES## -> insert all column names in UPPER CASE (3) ##KEYFIELDSAND- -> array of ints defining which cols are SORTORDER keys (4) ##tableobjectname## -> all lower case normalised table name

h e b b g e e e f c e b e ge

(5) ##ObjectName## -> upper and lower case normalised or specified object name (6) ##objectname## -> lower case normalised or specified object name (7) ##BaseTableObjects## -> enumerate all base table objects (8) ##inheritanceblock## -> recursively invoke parseSkeletonRecord() until ##endinheritanceblock## is encountered in the input template stream (9) ##index## -> insert an index counter, scoped within a given ##codeblock## (10) ##AttributeName## -> attribute name as a java-style class- - first byte upper case (11) ##attributeName## -> attribute name as a java-style method-- first byte lower case (12) ##ATTRIBUTENAME## -> UPPER CASE attribute name (13) ##attributeblock## -> recursively invoke parseSkeletonRecord() until ##endattributeblock## is encountered in the input template stream (14) ##attributeonlyblock## -> same as an ##attributeblock## but with no key fields (15) ##allkeyattributeblock# -> same as an ##attributeblock## but # with only key fields (16) ##keyFields## -> insert key fields as java-style method-- first byte lower case (17) ##MAXKEYCOUNT## -> insert nonnegative numeric integer constant of all object keys (18) ##ATTRIBUTECOUNT## -> insert nnic of count of attributes of object, including keys (19) ##parentKeyFields## -> insert key fields of top-level table object ONLY-- first byte lower case (20) ##attributesNoKeys## -> insert attribute names only, no primary or secondary keyfields (21) ##attributeNamesKeysQua -> all attributes, but at the end of a lfied## key field append keysuffix_ (22) ##keymap## -> insert metadata about key fields of underlying base tables (23) ##OBJECTNAME## -> UPPERCASE normalised or specified object name (24) ##counter+init## -> special tag to initialise a special internal counter. No output. (25) ##counter## -> insert the current value of the above counter, then increment (26) ##registryentrycount## -> insert the count of registry entries written (27) ##allcolumnblock## -> recursively invoke parseSkeletonRecord() until ##endcolumnblock## is encountered in the input template stream (28) ##COLUMNNAME## -> recursively insert a singular column name in UPPER CASE (29) ##TABLE## -> recursively insert a singular table name in UPPER CASE (30) ##entrycount++## -> increment registry entry count-- no output (31) ##allattributeblock## -> recursively invoke parseSkeletonRecord() until ##endcolumnblock## is encountered in the input template stream (32) ##DEFAULTMIN## -> based on datatype and attribute length, insert a reasonable default minimum value (33) ##DEFAULTMAX## -> based on datatype and attribute length, insert a reasonable default minimum value (34) ##VALIDATIONTYPE## -> based on datatype insert the validation type as defined in the OSFRulesObject base class (35) ##fieldlength## -> insert the maximum field length (36) ##picklistcandidates## -> insert picklist candidates from table scan or default string (37) ##iso639language## -> insert the current two byte iso639 language string (38) ##LANGUAGE## -> insert the current language descriptor (39) ##AttributeNameExpanded -> add a space before the 2nd through n capitals in an attribute name and then insert (40) ##language## -> insert the current language descriptor, in lower case (41) ##picklistvalues## -> insert all picklist values (multiple lines) or if no picklist exists for this column, suppress output of the record (42) ##picklistvalue## -> insert a unique picklist value guraranteed to be unique (43) ##picklistvalues## -> insert all unique picklist values (multiple lines) or if no picklist exists for this column, suppress output of the record (44) ##databaseblock## -> recursively invoke parseSkeletonRecord() until ##enddatabaseblock## is encountered in the input template stream, setting currentdatabase_ on each interation for each instance on the OSFDatabase list (45) ##DBLOGICALNAME## -> insert in upper case the intenral logical name of the currentdatabase_ (46) ##DBOWNER## -> insert in upper case the ownername of the currentdatabase_ and continue with further replacements (47) ##DBPASSWORD## -> insert in the case entered the password of the owner in the currentdatabase_ object and continue on with further replacements (48) ##DBTYPENAME## -> insert in upper case the jdbtools type name of the currentdatabase_, carry on with further replacements (49) ##DBSERVER## -> insert in the case entered the hostname or IP address in the currentdatabase_, carry on with further replacements (50) ##DBPORT## -> insert IP connect port in the currentdatabase_ object, carry on with further replacements (51) ##DBINSTANCE## -> insert in the case entered the instance name or SID in the currentdatabase_ object, carry on with further replacements (52) ##DBOWNER## -> insert in the case entered by the user the owner / user name in the

h e b b g e e e f c e b e ge

currentdatabase_ object, carry on with further replacements (53) ##MINKEYCOUNT## -> insert count of keys for a partially qualified read = key count of top level parent (54) ##hasparentconstraint## -> table is part of a relation / has a parent or owning table (55) ##testvalues## -> based upon current object context, insert a list of test attribute values (56) */ ##attributename## -> attribute name as an automatic declaration (57) ##attributenamekeysqual -> all attributes, lower case, at the ified## end of a key field append a lower case keysuffix_ (58) ##javadatatype## -> insert an appropriate Java data type depending on the normalised internal datatype (59) ##initializer## -> insert an appropriate initialiser depending on the normalised internal datatype (60) ##JavaPrimitiveObject## -> insert an name suitable for use in conversion methods (61) ##INTERNALDATA- -> insert the internal datatypes based TYPES## on the current _table (62)

Previous Doc Next Doc Go to Doc#

h e b b g e e e f c e b e ge

L4: Entry 2 of 3 File: USPT Jul 8, 2003

DOCUMENT-IDENTIFIER: US 6591272 B1
TITLE: Method and apparatus to make and transmit objects from a database on a server computer to a client computer

Detailed Description Text (187):
See the appendix for an example list of the special tag targets used by OSFGenerate to create persistent relational objects.

Detailed Description Text (523):
The above <title> tag sets the caption of window frame title. An additional title is desired within the frame and a hidden field is needed to verify the source of the input to the servlet:

Detailed Description Text (528):
The ##AttributeNameExpanded## tag takes the current attribute name and inserts a space before each capital letter. ##attributename## takes the current attribute name, converts it to lower case and inserts it into the output file. The name of the output file is DepartmentEmployeeEdit.html.

Detailed Description Paragraph Table (110):
"aa", "Afar", "ab", "Abkhazian", "af", "Afrikaans", "am", "Amharic", "ar", "Arabic", "as", "Assamese", "ay", "Aymara", "az", "Azerbaijani", "ba", "Bashkir", "be", "Byelorussian", "bg", "Bulgarian", "bh", "Bihari", "bi", "Bislama", "bn", "Bengali", "bo", "Tibetan", "br", "Breton", "ca", "Catalan", "co", "Corsican", "cs", "Czech", "cy", "Welsh", "da", "Danish", "de", "German", "dz", "Bhutani", "el", "Greek", "en", "English", "eo", "Esperanto", "es", "Spanish", "et", "Estonian", "eu", "Basque", "fa", "Persian", "fi", "Finnish", "fj", "Fiji", "fo", "Faroese", "fr", "French", "fy", "Frisian", "ga", "Irish", "gd", "Scots Gaelic", "gl", "Galician", "gn", "Guarani", "gu", "Gujarati", "ha", "Hausa", "he", "Hebrew", "hi", "Hindi", "hr", "Croatian", "hu", "Hungarian", "hy", "Armenian", "ia", "Interlingua", "id", "Indonesian", "ie", "Interlingue", "ik", "Inupiak", "is", "Icelandic", "it", "Italian", "iu", "Inuktitut", "ja", "Japanese", "jw", "Javanese", "ka", "Georgian", "kk", "Kazakh", "kl", "Greenlandic", "km", "Cambodian", "kn", "Kannada", "ko", "Korean", "ks", "Kashmiri", "ku", "Kurdish", "ky", "Kirghiz", "la", "Latin", "ln", "Lingala", "lo", "Laothian", "lt", "Lithuanian", "lv", "Latvian", "mg", "Malagasy", "mi", "Maori", "mk", "Macedonian", "ml", "Malayalam", "mn", "Mongolian", "mo", "Moldavian", "mr", "Marathi", "ms", "Malay", "mt", "Maltese", "my", "Burmese", "na", "Nauru", "ne", "Nepali", "nl", "Dutch", "no", "Norwegian", "oc", "Occitan", "om", "(Afan) Oromo", "or", "Oriya", "pa", "Punjabi", "pl", "Polish", "ps", "Pashto, Pushto", "pt", "Portuguese", "qu", "Quechua", "rm", "Rhaeto-Romance", "rn", "Kirundi", "ro", "Romanian", "ru", "Russian", "rw", "Kinyarwanda", "sa", "Sanskrit", "sd", "Sindhi", "sg", "Sangho", "sh", "Serbo-Croatian", "si", "Sinhalese", "sk", "Slovak", "sl", "Slovenian", "sm", "Samoan", "sn", "Shona", "so", "Somali", "sq", "Albanian", "sr", "Serbian", "ss", "Siswati", "st", "Sesotho", "su", "Sundanese", "sv", "Swedish", "sw", "Swahili", "ta", "Tamil", "te", "Telugu", "tg", "Tajik", "th", "Thai", "ti", "Tigrinya", "tk", "Turkmen", "tl", "Tagalog", "tn", "Setswana", "to", "Tonga", "tr", "Turkish", "ts", "Tsonga", "tt", "Tatar", "tw", "Twi", "ug", "Uighur", "uk", "Ukrainian", "ur", "Urdu", "uz", "Uzbek", "vi", "Vietnamese", "vo", "Volapuk", "wo", "Wolof", "xh", "Xhosa", "yi", "Yiddish", "yo", "Yoruba", "za", "Zhuang", "zh", "Chinese", "zu",

h e b b g e e e f c e b e ge

"Zulu",

Detailed Description Paragraph Table (111):
Target in Skeleton Function and Operation by Template file OSFGenerate ##Package## -> package target (0) ##TableObjectName## -> normalised table name (1) ##TABLENAME## -> insert table name in UPPER CASE (2) ##COLUMNNAMES## -> insert all column names in UPPER CASE (3) ##KEYFIELDSAND- -> array of ints defining which cols are SORTORDER keys (4) ##tableobjectname## -> all lower case normalised table name (5) ##ObjectName## -> upper and lower case normalised or specified object name (6) ##objectname## -> lower case normalised or specified object name (7) ##BaseTableObjects## -> enumerate all base table objects (8) ##inheritanceblock## -> recursively invoke parseSkeletonRecord() until ##endinheritanceblock## is encountered in the input template stream (9) ##index## -> insert an index counter, scoped within a given ##codeblock## (10) ##AttributeName## -> attribute name as a java-style class- - first byte upper case (11) ##attributeName## -> attribute name as a java-style method-- first byte lower case (12) ##ATTRIBUTENAME## -> UPPER CASE attribute name (13) ##attributeblock## -> recursively invoke parseSkeletonRecord() until ##endattributeblock## is encountered in the input template stream (14) ##attributeonlyblock## -> same as an ##attributeblock## but with no key fields (15) ##allkeyattributeblock# -> same as an ##attributeblock## but # with only key fields (16) ##keyFields## -> insert key fields as java-style method-- first byte lower case (17) ##MAXKEYCOUNT## -> insert nonnegative numeric integer constant of all object keys (18) ##ATTRIBUTECOUNT## -> insert nnic of count of attributes of object, including keys (19) ##parentKeyFields## -> insert key fields of top-level table object ONLY-- first byte lower case (20) ##attributesNoKeys## -> insert attribute names only, no primary or secondary keyfields (21) ##attributeNamesKeysQua -> all attributes, but at the end of a lfied## key field append keysuffix_ (22) ##keymap## -> insert metadata about key fields of underlying base tables (23) ##OBJECTNAME## -> UPPERCASE normalised or specified object name (24) ##counter+init## -> special tag to initialise a special internal counter. No output. (25) ##counter## -> insert the current value of the above counter, then increment (26) ##registryentrycount## -> insert the count of registry entries written (27) ##allcolumnblock## -> recursively invoke parseSkeletonRecord() until ##endcolumnblock## is encountered in the input template stream (28) ##COLUMNNAME## -> recursively insert a singular column name in UPPER CASE (29) ##TABLE## -> recursively insert a singular table name in UPPER CASE (30) ##entrycount++## -> increment registry entry count-- no output (31) ##allattributeblock## -> recursively invoke parseSkeletonRecord() until ##endcolumnblock## is encountered in the input template stream (32) ##DEFAULTMIN## -> based on datatype and attribute length, insert a reasonable default minimum value (33) ##DEFAULTMAX## -> based on datatype and attribute length, insert a reasonable default minimum value (34) ##VALIDATIONTYPE## -> based on datatype insert the validation type as defined in the OSFRulesObject base class (35) ##fieldlength## -> insert the maximum field length (36) ##picklistcandidates## -> insert picklist candidates from table scan or default string (37) ##iso639language## -> insert the current two byte iso639 language string (38) ##LANGUAGE## -> insert the current language descriptor (39) ##AttributeNameExpanded -> add a space before the 2nd through n capitals in an attribute name and then insert (40) ##language## -> insert the current language descriptor, in lower case (41) ##picklistvalues## -> insert all picklist values (multiple lines) or if no picklist exists for this column, suppress output of the record (42) ##picklistvalue## -> insert a unique picklist value gurarantеed to be unique (43) ##picklistvalues## -> insert all unique picklist values (multiple lines) or if no picklist exists for this column, suppress output of the record (44) ##databaseblock## -> recursively invoke parseSkeletonRecord() until ##enddatabaseblock## is encountered in the input template stream, setting currentdatabase_ on each interation for each instance on the OSFDatabase list (45) ##DBLOGICALNAME## -> insert in upper case the intenral logical name of the currentdatabase_ (46) ##DBOWNER## -> insert in upper case the ownername of the currentdatabase_ and continue with further replacements (47) ##DBPASSWORD## -> insert in the case entered the password of the owner in the currentdatabase_ object

and continue on with further replacements (48) ##DBTYPENAME## -> insert in upper case the jdbtools type name of the currentdatabase_, carry on with further replacements (49) ##DBSERVER## -> insert in the case entered the hostname or IP address in the currentdatabase_, carry on with further replacements (50) ##DBPORT## -> insert IP connect port in the currentdatabase_ object, carry on with further replacements (51) ##DBINSTANCE## -> insert in the case entered the instance name or SID in the currentdatabase_ object, carry on with further replacements (52) ##DBOWNER## -> insert in the case entered by the user the owner / user name in the currentdatabase_ object, carry on with further replacements (53) ##MINKEYCOUNT## -> insert count of keys for a partially qualified read = key count of top level parent (54) ##hasparentconstraint## -> table is part of a relation / has a parent or owning table (55) ##testvalues## -> based upon current object context, insert a list of test attribute values (56) */ ##attributename## -> attribute name as an automatic declaration (57) ##attributenamekeysqual -> all attributes, lower case, at the ified## end of a key field append a lower case keysuffix_ (58) ##javadatatype## -> insert an appropriate Java data type depending on the normalised internal datatype (59) ##initializer## -> insert an appropriate initialiser depending on the normalised internal datatype (60) ##JavaPrimitiveObject## -> insert an name suitable for use in conversion methods (61) ##INTERNALDATA- -> insert the internal datatypes based TYPES## on the current _table (62)

Detailed Description Paragraph Table (112):

```
/* * @(#)JobEmployee_fr.java 1.0 */ /** resource bundle for object: JobEmployee,
language: French note: use the french.sh sed script to update this and all resource
bundles for the French language. */ package com.tricoron.osftest.client; import
java.util.ListResourceBundle; public class JobEmployee_fr extends
ListResourceBundle { static final Object[ ] [ ] contents = { // Field / attribute
descriptors { "JobID", "French: Job I D"}, { "Function", "French: Function"},
{ "EmployeeID", "French: Employee I D"}, { "LastName", "French: Last Name"},
{ "FirstName", "French: First Name"}, { "MiddleInitial", "French: Middle Initial"},
{ "ManagerID", "French: Manager I D"}, { "HireDate", "French: Hire Date"},
{ "Salary", "French: Salary"}, { "Commission", "French: Commission"},
{ "DepartmentID", "French: Department I D"}, // Picklist value descriptors
{ "JOB$JOB_ID$669", "French: 669:"}, { "JOB$JOB_ID$668", "French: 668:"},
{ "JOB$JOB_ID$672", "French: 672:"}, { "JOB$JOB_ID$671", "French: 671:"},
{ "JOB$JOB_ID$670", "French: 670:"}, { "EMPLOYEE$DEPARTMENT_ID$43", "French: 43:"},
{ "EMPLOYEE$DEPARTMENT_ID$34", "French: 34:"}, { "EMPLOYEE$DEPARTMENT_ID$30".
"French: 30:"}, { "EMPLOYEE$DEPARTMENT_ID$24", "French: 24:"},
{ "EMPLOYEE$DEPARTMENT_ID$23", "French: 23:"}, { "EMPLOYEE$DEPARTMENT_ID$20",
"French: 20:"}, { "EMPLOYEE$DEPARTMENT_ID$14", "French: 14:"},
{ "EMPLOYEE$DEPARTMENT_ID$13", "French: 13:"}, { "EMPLOYEE$DEPARTMENT_ID$12",
"French: 12:"}, { "EMPLOYEE$DEPARTMENT_ID$10", "French: 10:"}, // tag to return
"missing" in the event a null key is passed to getString( ) { " ", "missing"} };
public Object[ ] [ ] getContents( ) { return contents; } }
```

Previous Doc Next Doc Go to Doc#

h e b b g e e e f c e b e ge

9/840,655 – 4/23/01

Results of Search in 1976 to present db for:
(((SPEC/rule AND ACLM/metadata) AND (SPEC/contract OR SPEC/agree)) AND SPEC/XML): 3 patents.

1 6,640,145 T Media recording device with packet data interface

2 6,591,272 T Method and apparatus to make and transmit objects from a database on a server computer to a client computer

3 5,970,490 T Integration platform for heterogeneous databases

6640145

4. The intelligent media device according to claim 1, wherein said digital memory stores ***metadata*** relating to said received audio and/or video data.
19. The intelligent media device according to claim 1,

wherein:

said packet data communications interface communicates with a local area network to at least one external machine and communicates with a remote human user in a manner suitable for use with a browser, to provide instructions for said intelligent server, said local area network optionally including a wireless communications link;

the at least one external machine comprising one or more selected from the group consisting of a telephony system, an imaging system, a videoconferencing system, a consumer electronic device, a security system, an alarm system, an environmental control system, an automobile; an illumination system, a domestic appliance; a pump, and a flow control;

said digital memory further storing ***metadata*** relating to said received audio and/or video data;

said intelligent server having one or more characteristics selected from the following:

being adaptive to at least one of a human user and/and an external machine,

having a mode of operation for acting autonomously based on an inferred program,

having a mode of operation for receiving feedback from a human user;

being adapted to analyze communications received from a human user for likely error,

being adapted to translate a native external machine interface for presentation to the human user, and

being adapted to receive and execute general purpose computer instructions,

further comprising a digital rights manager for enforcing a set of externally supplied restrictions associated with said received audio and/or video data, and a cryptographic processor for selectively cryptoprocessing audio and/or video data in dependence on said rights manager.

6591272

(3. A method of communicating elements of a database, having a ***metadata,*** between a server computer and a client computer, comprising:

generating a pseudo-object by said server computer, said pseudo-object comprising data of said elements;

generating ***metadata*** of said elements, wherein said ***metadata*** is relationship of said data of said elements, wherein said generating step further comprising:

reading said ***metadata*** of said database;

translating a list of said ***metadata*** to a standardized view of said database;

accessing skeleton code templates representative of final classes to be produced;

generating source code for the classes of the objects desired, and scripts to compile said classes into executable form;

producing said object desired by enveloping said data and ***metadata***;

transmitting said pseudo-object and ***metadata*** from said server computer to said client computer; and

assembling said elements to final distributed objects by said client computer from said pseudo-object and ***metadata*** received.

4. The method of claim 3 wherein said reading step further comprises:

generating a pick list based upon an inversion of said database table; and

generating a script containing all unique foreign language strings to be translated.

5. The method of claim 4 wherein said assembling step further comprises:

translating said script to a given language.

6. An article of manufacture comprising:

a computer usable medium having computer readable program code embodied therein configured to translate ***metadata*** of a database into objects desired, the computer readable program code in said article of manufacture comprising:

computer readable program code configured to cause a computer to read said ***metadata*** of said database to determine characteristics of said database and their relationship, wherein said computer readable program code further comprises;

computer readable program code configured to generate a pick list based upon an inversion of elements of said database; and

computer readable program code configured to generate a script containing all unique foreign language strings to be translated;

computer readable program code configures to cause a computer to assemble a list of the ***metadata*** of said database and their relationship in a pseudo-standardized view of said database;

computer readable program code configured to cause a computer to read skeleton code templates representative of final classes to be produced;

computer readable program code configured to cause a computer to generate source code for the class of the object desired; and

computer readable program code configured to cause a computer to produce said objects desired by enveloping

values assembled in said templates.

7. An article of manufacture comprising:

a computer usable medium having computer readable program code embodied therein configured to communicate elements of a database table between a server computer and a client computer, the computer readable program code in said article of manufacture comprising:

computer readable program code configured to generate a pseudo-object by said server computer, said pseudo-object comprising data of said elements and to generate ***metadata*** of said elements, wherein said ***metadata*** is relationship of said data, wherein said computer readable program code configured to generate an object by said server computer further comprising:

a computer usable medium having computer readable program code embodied therein configured to translate elements of a relational database table into objects desired, the computer readable program code in said article of manufacture comprising:

computer readable program code configured to cause a computer to read said elements of said database table to determine values of said elements and their relationship;

computer readable program code configured to cause a computer to assemble a list of the values of said elements and their relationship to a standardized view of said database table;

computer readable program code configured to cause a computer to access skeleton code templates representative of final objects to be produced;

computer readable program code configured to cause a computer to generate source code for the class of the object desired; and

computer readable program code configured to cause a computer to produce said objects desired by enveloping values assembled in one of said templates;

computer readable program code configured to transmit said pseudo-object and said ***metadata*** from said server computer to said client computer; and

computer readable program code configured to assemble said elements by said client computer from said object and ***metadata*** received.)

5,970,490
(2. The method as claimed in claim 1, wherein the first high level data structure specification and second high level data structure specification comprise:

diverse metatadata, including semantic ***metadata***.

10. The method as claimed in claim 9, wherein the first high level data structure specification and second high level data structure specification comprise: diverse ***metadata,*** including semantic ***metadata***.

At the structural level, the primary characteristics of SGML (Standard Generalized Markup Language), HTML (Hypertext Markup Language), and the new evolving ***XML*** (Extensible Markup Language) have been subsumed, as well as the heterogeneous database structures--including relational and object-oriented models, and to provide extensibility to address multimedia data.

XML is an emerging language called the Extensible Markup Language. It is intended to extend and eventually supersede HTML (Hypertext Markup Language), and to be in the spirit of SGML (Standard Generalized Markup Language) but to be substantially simpler than SGML.

When the SEMDAL language is compared with ***XML*** it can be shown that: 1) ***XML*** capabilities are subsumed, 2) SEMDAL is much more concise, and 3) semantic representational capabilities are provided that go beyond the natural abilities of ***XML***.

First the SEMIDAL representation is presented, followed by the ***XML*** representation given by ***XML*** advocates. The example is that of a bookstore order system, which sells books, records, and coffee. The example comes from ***XML*** literature. The example is reexpressed in the structure SEMDAL and meaningful semantic attribute information has been added--these semantics are not present in the ***XML*** version given in earlier even though that specification is much more verbose)

L3: Entry 1 of 2 File: USPT Jul 8, 2003

DOCUMENT-IDENTIFIER: US 6591272 B1
TITLE: Method and apparatus to make and transmit objects from a database on a server computer to a client computer

Detailed Description Paragraph Table (111):
Target in Skeleton Function and Operation by Template file OSFGenerate ##Package## -> package target (0) ##TableObjectName## -> normalised table name (1) ##TABLENAME## -> insert table name in UPPER CASE (2) ##COLUMNNAMES## -> insert all column names in UPPER CASE (3) ##KEYFIELDSAND- -> array of ints defining which cols are SORTORDER keys (4) ##tableobjectname## -> all lower case normalised table name (5) ##ObjectName## -> upper and lower case normalised or specified object name (6) ##objectname## -> lower case normalised or specified object name (7) ##BaseTableObjects## -> enumerate all base table objects (8) ##inheritanceblock## -> recursively invoke parseSkeletonRecord() until ##endinheritanceblock## is encountered in the input template stream (9) ##index## -> insert an index counter, scoped within a given ##codeblock## (10) ##AttributeName## -> attribute name as a java-style class- - first byte upper case (11) ##attributeName## -> attribute name as a java-style method-- first byte lower case (12) ##ATTRIBUTENAME## -> UPPER CASE attribute name (13) ##attributeblock## -> recursively invoke parseSkeletonRecord() until ##endattributeblock## is encountered in the input template stream (14) ##attributeonlyblock## -> same as an ##attributeblock## but with no key fields (15) ##allkeyattributeblock# -> same as an ##attributeblock## but # with only key fields (16) ##keyFields## -> insert key fields as java-style method-- first byte lower case (17) ##MAXKEYCOUNT## -> insert nonnegative numeric integer constant of all object keys (18) ##ATTRIBUTECOUNT## -> insert nnic of count of attributes of object, including keys (19) ##parentKeyFields## -> insert key fields of top-level table object ONLY-- first byte lower case (20) ##attributesNoKeys## -> insert attribute names only, no primary or secondary keyfields (21) ##attributeNamesKeysQua -> all attributes, but at the end of a lfied## key field append keysuffix_ (22) ##keymap## -> insert metadata about key fields of underlying base tables (23) ##OBJECTNAME## -> UPPERCASE normalised or specified object name (24) ##counter+init## -> special tag to initialise a special internal counter. No output. (25) ##counter## -> insert the current value of the above counter, then increment (26) ##registryentrycount## -> insert the count of registry entries written (27) ##allcolumnblock## -> recursively invoke parseSkeletonRecord() until ##endcolumnblock## is encountered in the input template stream (28) ##COLUMNNAME## -> recursively insert a singular column name in UPPER CASE (29) ##TABLE## -> recursively insert a singular table name in UPPER CASE (30) ##entrycount++## -> increment registry entry count-- no output (31) ##allattributeblock## -> recursively invoke parseSkeletonRecord() until ##endcolumnblock## is encountered in the input template stream (32) ##DEFAULTMIN## -> based on datatype and attribute length, insert a reasonable default minimum value (33) ##DEFAULTMAX## -> based on datatype and attribute length, insert a reasonable default minimum value (34) ##VALIDATIONTYPE## -> based on datatype insert the validation type as defined in the OSFRulesObject base class (35) ##fieldlength## -> insert the maximum field length (36) ##picklistcandidates## -> insert picklist candidates from table scan or default string (37) ##iso639language## -> insert the current two byte iso639 language string (38) ##LANGUAGE## -> insert the current language descriptor (39) ##AttributeNameExpanded -> add a space before the 2nd through n capitals in an attribute name and then insert (40) ##language## -> insert the current language

descriptor, in lower case (41) ##picklistvalues## -> insert all picklist values (multiple lines) or if no picklist exists for this column, suppress output of the record (42) ##picklistvalue## -> insert a unique picklist value guraranteed to be unique (43) ##picklistvalues## -> insert all unique picklist values (multiple lines) or if no picklist exists for this column, suppress output of the record (44) ##databaseblock## -> recursively invoke parseSkeletonRecord() until ##enddatabaseblock## is encountered in the input template stream, setting currentdatabase_ on each interation for each instance on the OSFDatabase list (45) ##DBLOGICALNAME## -> insert in upper case the intenral logical name of the currentdatabase_ (46) ##DBOWNER## -> insert in upper case the ownername of the currentdatabase_ and continue with further replacements (47) ##DBPASSWORD## -> insert in the case entered the password of the owner in the currentdatabase_ object and continue on with further replacements (48) ##DBTYPENAME## -> insert in upper case the jdbtools type name of the currentdatabase_, carry on with further replacements (49) ##DBSERVER## -> insert in the case entered the hostname or IP address in the currentdatabase_, carry on with further replacements (50) ##DBPORT## -> insert IP connect port in the currentdatabase_ object, carry on with further replacements (51) ##DBINSTANCE## -> insert in the case entered the instance name or SID in the currentdatabase_ object, carry on with further replacements (52) ##DBOWNER## -> insert in the case entered by the user the owner / user name in the currentdatabase_ object, carry on with further replacements (53) ##MINKEYCOUNT## -> insert count of keys for a partially qualified read = key count of top level parent (54) ##hasparentconstraint## -> table is part of a relation / has a parent or owning table (55) ##testvalues## -> based upon current object context, insert a list of test attribute values (56) */ ##attributename## -> attribute name as an automatic declaration (57) ##attributenamekeysqual -> all attributes, lower case, at the ified## end of a key field append a lower case keysuffix_ (58) ##javadatatype## -> insert an appropriate Java data type depending on the normalised internal datatype (59) ##initializer## -> insert an appropriate initialiser depending on the normalised internal datatype (60) ##JavaPrimitiveObject## -> insert an name suitable for use in conversion methods (61) ##INTERNALDATA- -> insert the internal datatypes based TYPES## on the current _table (62)

Previous Doc Next Doc Go to Doc#

h e b b g e e e f c e h e e ge

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#
End of Result Set

Generate Collection Print

L3: Entry 2 of 2 File: USPT Oct 19, 1999

DOCUMENT-IDENTIFIER: US 5970490 A
TITLE: Integration platform for heterogeneous databases

Detailed Description Text (77):
Referring again the FIG. 2, the HLDSS is parsed and processed by schema analyzers 32 and 52 to create annotated logical structure diagrams (LSDs) 34 and 54, each of which is a schematic structure graph (like a parse tree) that represents the schema and data structures of the data resources. Logical structure diagrams provide a form of meta-schema in that LSD's may be used to graphically describe different data models and schemas. The motivation for the LSD internal representation is that a context-independent uniform graph-based representation can represent all anticipated data resource schemata and structures. The context-dependent interpretation is dictated by annotations, which are processed by the HLDSS preprocessor and the recognizer generator 36 and view generator 56 modules.

Previous Doc Next Doc Go to Doc#

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#

Generate Collection Print

L2: Entry 2 of 3 File: USPT Jul 8, 2003

DOCUMENT-IDENTIFIER: US 6591272 B1
TITLE: Method and apparatus to make and transmit objects from a database on a server computer to a client computer

Detailed Description Text (356):
In contract to the rather thin OSFServerObject class, OSFServletObject has a lot to it. This is because we decided to not use a pre-built package of HTML page classes to build dynamic HTML streams (fact is we looked at a few and did not find one which satisfied our requirements). So we rolled our own. We are glad we did since the performance and flexibility we achieved was more than we expected.

Detailed Description Text (696):
OSFORBStream.DELETESTREAMs are bidirectional as well as they inform a client that an object for which the client has interest no longer exists or the subscription contract has terminated for that particular object.

Previous Doc Next Doc Go to Doc#